

课程回顾

■求解递归式（计算时间复杂度）：

➤代入法：猜测解+数学归纳法证明（注意边界条件！）

- 注意事项：做出好的猜测、细节修正、避免陷阱、变量代换

➤递归树法

➤主方法

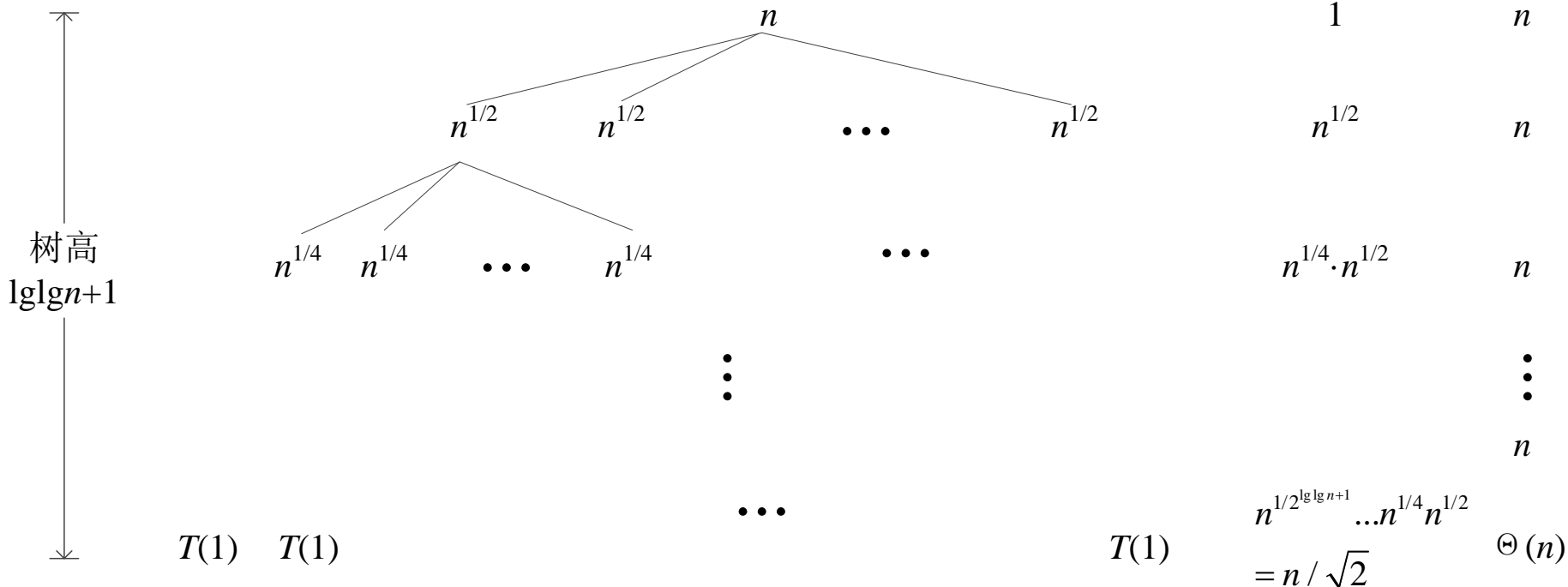
■递归式求解例（主方法 $\xrightarrow{\text{无法解决}}$ 代入法/递归树法）

递归式求解例 4-3 j

j. $T(n) = \sqrt{n}T(\sqrt{n}) + n$

解：（递归树法）

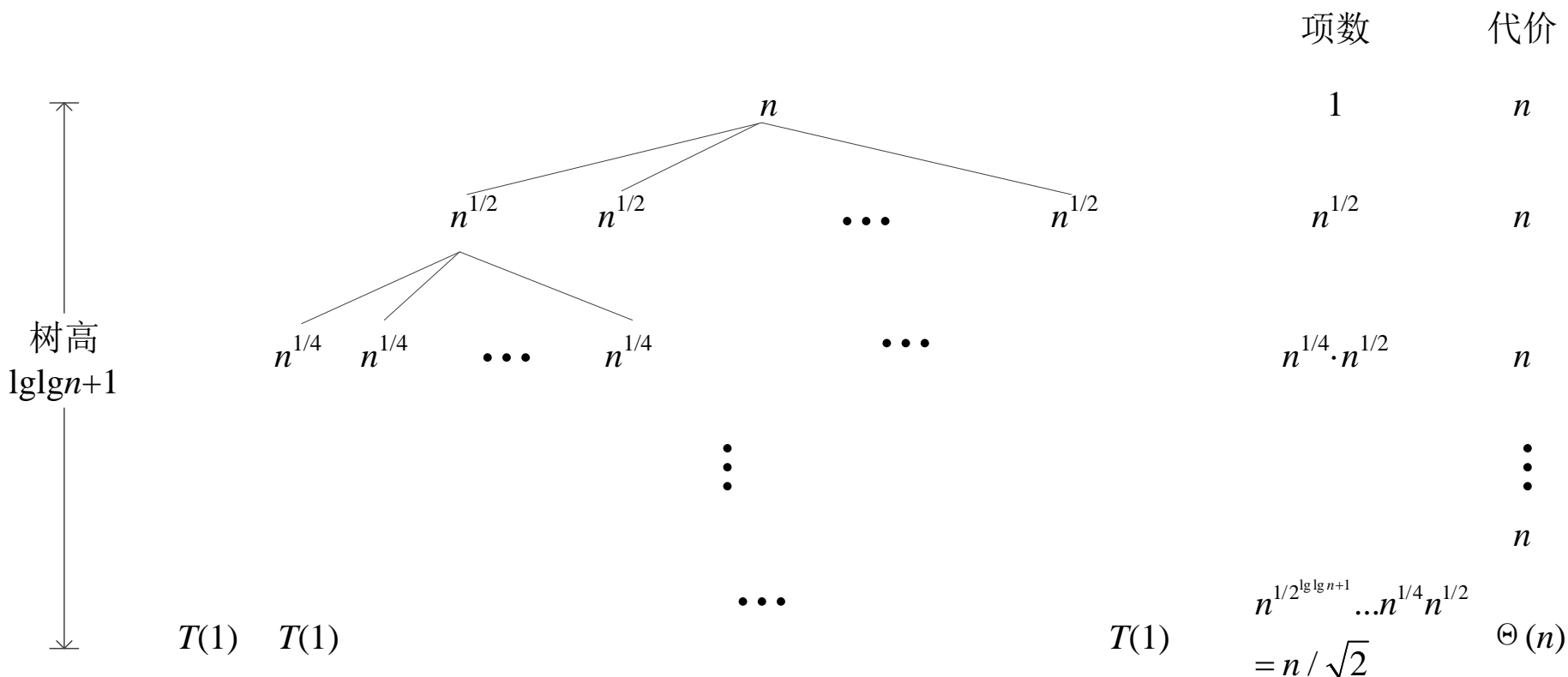
树高 i 为满足 $n^{1/2^i} < 2$ 的最小的整数 i



递归式求解例 4-3 j (续)

j. $T(n) = \sqrt{n}T(\sqrt{n}) + n$

解: $T(n) = n \lg \lg n + \Theta(n) = \Theta(n \lg \lg n)$



分治法的适用条件

1. 该问题的规模缩小到一定的程度就可以容易地解决
 - 因为问题的计算复杂性一般是随着问题规模的增加而增加，因此大部分问题满足这个特征
2. 该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质
 - 这条特征是应用分治法的前提，它也是大多数问题可以满足的，此特征反映了递归思想的应用

分治法的适用条件 (续)

3. 利用该问题分解出的子问题的解可以合并为该问题的解

- 能否利用分治法完全取决于问题是否具有这条特征，如果具备了前两条特征，而不具备第三条特征，则可以考虑贪心算法或动态规划

4. 该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子问题

- 这条特征涉及到分治法的效率，如果各子问题是不独立的，则分治法要做许多不必要的工作，重复地解公共的子问题，此时虽然也可用分治法，但一般用动态规划较好

分治法求解——二分搜索

■ 给定已按升序排好序的 n 个元素 $A[1..n]$ ，现要在这 n 个元素中找出一特定元素 x

■ 适用条件分析：

➤ 该问题的规模缩小到一定的程度就可以容易地解决 ✓

- 分析：如果 $n=1$ 即只有一个元素，则只要比较这个元素和 x 就可以确定 x 是否在数组中

➤ 该问题可以分解为若干个规模较小的相同问题 ✓

- 分析：比较 x 和 A 的中间元素 $A[mid]$ ，若 $x=A[mid]$ ，则 x 在 A 中的位置就是 mid ；如果 $x<A[mid]$ ，由于 A 是递增排序的，因此假如 x 在 A 中的话， x 必然排在 $A[mid]$ 的前面，所以只要在 $A[low..mid-1]$ 查找 x 即可；如果 $x>A[mid]$ ，同理只要在 $A[mid+1..high]$ 查找 x 即可。无论如何查找 x ，其方法都和 A 中查找 x 一样，只不过是查找的规模缩小了

➤ 分解出的子问题的解可以合并为原问题的解 ✓

分治法求解——二分搜索 (续)

■ 给定已按升序排好序的 n 个元素 $A[1..n]$ ，现要在这 n 个元素中找出一特定元素 x

■ 适用条件分析：

➤ 分解出的各个子问题是相互独立的 ✓

- 分析：显然此问题分解出的子问题相互独立，即在 $A[low..mid-1]$ 和 $A[mid+1..high]$ 查找 x 是独立的子问题

分治法求解——二分搜索 (续)

■适用范围：顺序表、有序序列

■基本思想

1. 设 $A[low..high]$ 是当前查找区间，当 $low \leq high$ 时进行查找（否则 A 数组为空）
2. 确定中点位置 $mid = \lfloor (low + high) / 2 \rfloor$
3. 将待查值 x 与 $A[mid]$ 比较
 - ① 若 $x = A[mid]$ ：查找成功，返回位置 mid
 - ② 若 $x < A[mid]$ ：查找左子数组 $A[low..mid-1]$
 - ③ 若 $x > A[mid]$ ：查找右子数组 $A[mid+1..high]$

分治法求解——二分搜索 (续)

```
BINARY_SEARCH(A, x, low, high)
1  if low ≤ high
2    mid ← ⌊(low + high)/2⌋
3    if x = A[mid]
4      return mid
5    if x < A[mid]
6      return BINARY_SEARCH(A, x, low, mid-1)
7    else return BINARY_SEARCH(A, x, mid+1, high)
8  return 0
```

■最坏情况运行时间： $T(n)=T(n/2)+\Theta(1)$ ，由主方法易得 $T(n)=\Theta(\lg n)$

分治法求解——最大子数组问题

- （教材p39）寻找数组 $A[1..n]$ 中，和最大的非空连续子数组



图 4-3 股票价格变化值的最大子数组问题。本例中，子数组 $A[8..11]$ 的和是 43，是 A 的所有连续子数组中和最大的

- 暴力求解：穷举所有非空连续子数组，即穷举所有可能的起始、终止位置： $C_n^2 + \boxed{n} = \frac{n(n-1)}{2} + n = \Theta(n^2)$
起始终止位置相同
- 可在 $O(n^3)$ 时间解决，改进方法可在 $O(n^2)$ 时间解决

分治法求解——最大子数组问题 (续)

■分治法适用条件分析：

➤该问题的规模缩小到一定程度就可容易解决 ✓

- 分析：如果 $n=1$ 即只有一个元素，只要输出该元素值即可

➤该问题可以分解为若干个规模较小的相同问题 ✓

- 分析：将数组 $A[low..high]$ 对半划分，中间位置为 mid ， A 的非空连续子数组 $A[i..j]$ 肯定在三种情况之一：

1. 完全位于子数组 $A[low..mid]$ 中： $low \leq i \leq j \leq mid$
2. 完全位于子数组 $A[mid+1..high]$ 中： $mid+1 \leq i \leq j \leq high$
3. 跨越了中点 mid ： $low \leq i \leq mid < j \leq high$

前两种情况相当于求解规模更小的原问题；第3种情况易于计算

分治法求解——最大子数组问题 (续)

■分治法适用条件分析：

➤分解出的子问题的解可以合并为原问题的解 ✓

- 分析：将三种情况得到的最大和进行比较即可

➤分解出的各个子问题是相互独立的 ✗

- 分析：求和过程可能多次计算，并非独立

存在更好的解法：动态规划、贪心算法
但分治法依旧可以实现

分治法求解——最大子数组问题 (续)

■ 寻找跨越中点 mid 的最大子数组

$A[\max_left..\max_right] : low \leq \max_left \leq mid < \max_right \leq high$

FIND_MAX_CROSSING_SUBARRAY($A, low, mid, high$)

1 $left_sum \leftarrow -\infty$

2 $sum \leftarrow 0$

3 **for** $i \leftarrow mid$ **downto** low **do**

4 $sum \leftarrow sum + A[i]$

5 **if** $sum > left_sum$

6 $left_sum \leftarrow sum; \max_left \leftarrow i$

7 $right_sum \leftarrow -\infty$

8 $sum \leftarrow 0$

9 **for** $j \leftarrow mid+1$ **to** $high$ **do**

10 $sum \leftarrow sum + A[j]$

11 **if** $sum > right_sum$

12 $right_sum \leftarrow sum; \max_right \leftarrow j$

13 **return** ($\max_left, \max_right, left_sum + right_sum$)

$\Theta(n)$

分治法求解——最大子数组问题 (续)

```
FIND_MAXIMUM_SUBARRAY(A, low, high)
1  if high = low
2    return (low, high, A[low])
3  else mid  $\leftarrow \lfloor (\textit{low} + \textit{high})/2 \rfloor$ 
4    (left_low, left_high, left_sum)  $\leftarrow$ 
      FIND_MAXIMUM_SUBARRAY(A, low, mid) // $T(n/2)$ 
5    (right_low, right_high, right_sum)  $\leftarrow$ 
      FIND_MAXIMUM_SUBARRAY(A, mid+1, high) // $T(n/2)$ 
6    (cross_low, cross_high, cross_sum)  $\leftarrow$ 
      FIND_MAX_CROSSING_SUBARRAY(A, low, mid, high) // $\Theta(n)$ 
7  if left_sum  $\geq$  right_sum and left_sum  $\geq$  cross_sum
8    return (left_low, left_high, left_sum)
9  elseif right_sum  $\geq$  left_sum and right_sum  $\geq$  cross_sum
10   return (right_low, right_high, right_sum)
11 else return (cross_low, cross_high, cross_sum)
```

分治法求解——最大子数组问题 (续)

■ 算法分析

$$T(n) = 2T(n/2) + \Theta(n)$$

➤ 与归并排序类似

➤ 主方法求解： $a=2$, $b=2$, $f(n)=cn$, $n^{\log_b a} = n^{\log_2 2} = \Theta(n)$

➤ $f(n) = \Theta(n)$

➤ $n^{\log_b a}$ 与 $f(n)$ 同级别，主定理第2种情况： $T(n)=\Theta(n \lg n)$

分治法求解——矩阵乘法

■ 方阵乘法问题：设矩阵 $A=(a_{ij})$ 和 $B=(b_{ij})$ 都是 $n \times n$ 的方阵，则定义乘积矩阵 $C=A \cdot B$ 中的元素 c_{ij} 为：

$$c_{ij} = \sum_{k=1}^n a_{ik} \cdot b_{kj}$$

■ 传统方法： $O(n^3)$

➤ 计算每一个元素 c_{ij} 都要做 n 次乘法和 $n-1$ 次加法，因此计算矩阵 C 所有 n^2 个元素所需时间为 $O(n^3)$

是否有更快的求解方法？考虑分治法

分治法求解——矩阵乘法 (续)

- 不妨假设 $n=2^k$ ，将矩阵A, B, C都分块成4个大小相等的子矩阵

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

- 此时

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$

$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$

$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$

$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

- 8次规模为 $n/2$ 的矩阵乘法，4次包含 $n^2/4$ 个元素的矩阵加法，时间复杂度： $T(n)=8T(n/2)+\Theta(n^2)=\Theta(n^3)$

减少划分的子问题数量，即降低矩阵乘法次数

分治法求解——Strassen算法

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

■将分块矩阵进行如下操作：

$$P_1 = A_{11}(B_{12} - B_{22})$$

$$P_2 = (A_{11} + A_{12})B_{22}$$

$$P_3 = (A_{21} + A_{22})B_{11}$$

$$P_4 = A_{22}(B_{21} - B_{11})$$

$$P_5 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$P_6 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$P_7 = (A_{11} - A_{21})(B_{11} + B_{12})$$

$$\begin{aligned} C_{11} &= P_5 + P_4 - P_2 + P_6 \\ C_{12} &= P_1 + P_2 \\ C_{21} &= P_3 + P_4 \\ C_{22} &= P_5 + P_1 - P_3 - P_7 \end{aligned}$$

子问题数量降低为7, $T(n)=7T(n/2)+\Theta(n^2)=\Theta(n^{\lg 7})$

分治法求解——大整数乘法

■考虑两个 n 位大整数 $A=a_1a_2a_3\dots a_n$ 和 $B=b_1b_2b_3\dots b_n$ 相乘（ a_i 和 b_j 分别表示 A 第 i 位的数字和 B 第 j 位的数字），可列竖式计算：

$$\begin{array}{r} a_1 a_2 \dots a_n \\ \times \quad b_1 b_2 \dots b_n \\ \hline (d_{10}) d_{11} d_{12} \dots d_{1n} \\ (d_{20}) d_{21} d_{22} \dots d_{2n} \\ \dots\dots\dots \\ (d_{n0}) d_{n1} d_{n2} \dots d_{nn} \\ \hline \Sigma \end{array}$$

➤计算 n^2 次乘法和 $n-1$ 次加法，时间复杂度 $\Theta(n^2)$

分治法求解——大整数乘法 (续)

■与矩阵乘法类似考虑采用分治法解决：将一个 n 位大整数划分为两个 $n/2$ 位的大整数，即 $A=A_1A_2$ 和 $B=B_1B_2$ ($A_1=a_1a_2\dots a_{n/2}$, $A_2=a_{n/2+1}a_{n/2+2}\dots a_n$, $B_1=b_1b_2\dots b_{n/2}$, $B_2=b_{n/2+1}b_{n/2+2}\dots b_n$)

$$A \cdot B = A_1 \cdot B_1 \cdot 10^n + (A_1 \cdot B_2 + A_2 \cdot B_1) \cdot 10^{n/2} + A_2 \cdot B_2$$

➤例： $A=2135$, $B=4014$, 划分后 $A_1=21$, $A_2=35$, $B_1=40$, $B_2=14$

$$\begin{aligned} A \cdot B &= (21 \cdot 10^2 + 35) \cdot (40 \cdot 10^2 + 14) \\ &= 21 \cdot 40 \cdot 10^4 + (21 \cdot 14 + 35 \cdot 40) \cdot 10^2 + 35 \cdot 14 \end{aligned}$$

➤ $T(n)=4T(n/2)+\Theta(n)=\Theta(n^2)$ 没有改进，减少子问题数量！

分治法求解——大整数乘法 (续)

■将子问题数量从4降到3:

$$\triangleright C_1 = A_1 \cdot B_1, \quad C_2 = A_2 \cdot B_2, \quad C_3 = (A_1 + A_2) \cdot (B_1 + B_2)$$

$$\triangleright A \cdot B = C_1 \cdot 10^n + (C_3 - C_1 - C_2) \cdot 10^{n/2} + C_2$$

$$\triangleright T(n) = 3T(n/2) + \Theta(n) = \Theta(n^{\lg 3})$$

■如果将大整数分成更多段，用更复杂的方式把它们组合起来，将有可能得到更优的算法

■最终这个思想导致快速傅利叶变换(Fast Fourier Transform)的产生（FFT可参见教材Chap 30）

分治法求解——快速排序

■（教材p95）快速排序尽管最坏情况时间复杂度是 $\Theta(n^2)$ ，但平均情况时间复杂度为 $\Theta(n \lg n)$

■基于比较的排序算法时间下界为：

$$\begin{aligned}\lg(n!) &= \lg(\sqrt{2\pi n}(n/e)^n(1 + \Theta(1/n))) \\ &= \lg(2\pi n)/2 + n \lg(n/e) + \lg(1 + \Theta(1/n)) \\ &= \lg(2\pi)/2 + (\lg n)/2 + n \lg n - n \lg e + \lg(1 + \Theta(1/n)) \\ &\approx n \lg n - 1.44n + \Theta(\lg n)\end{aligned}$$

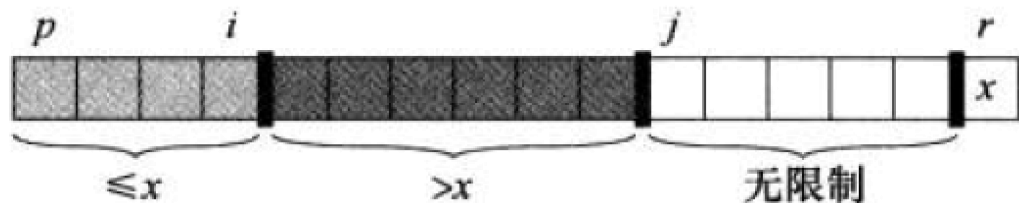
■快速排序平均情况： $1.39n \lg n + O(n)$ ，因系数较小称为“快排”

分治法求解——快速排序 (续)

■ 算法描述

QUICKSORT(A, p, r)

```
1 if  $p < r$ 
2    $q \leftarrow \text{PARTITION}(A, p, r)$ 
3   QUICKSORT( $A, p, q-1$ )
4   QUICKSORT( $A, q+1, r$ )
```



PARTITION(A, p, r)

```
1  $x \leftarrow A[r]$  //划分元/主元(pivot element)
2  $i \leftarrow p - 1$ 
3 for  $j \leftarrow p$  to  $r-1$  do
4   if  $A[j] \leq x$ 
5      $i \leftarrow i + 1$ ; exchange  $A[i]$  with  $A[j]$ 
6 exchange  $A[i+1]$  with  $A[r]$ 
7 return  $i+1$ 
```

循环不变式:

1. $A[p..i] \leq x$
2. $A[i+1..j-1] > x$
3. $A[r] = x$

该划分使得:

$$A[p..q-1] \leq A[q] < A[q+1..r]$$

分治法求解——快速排序 (续)

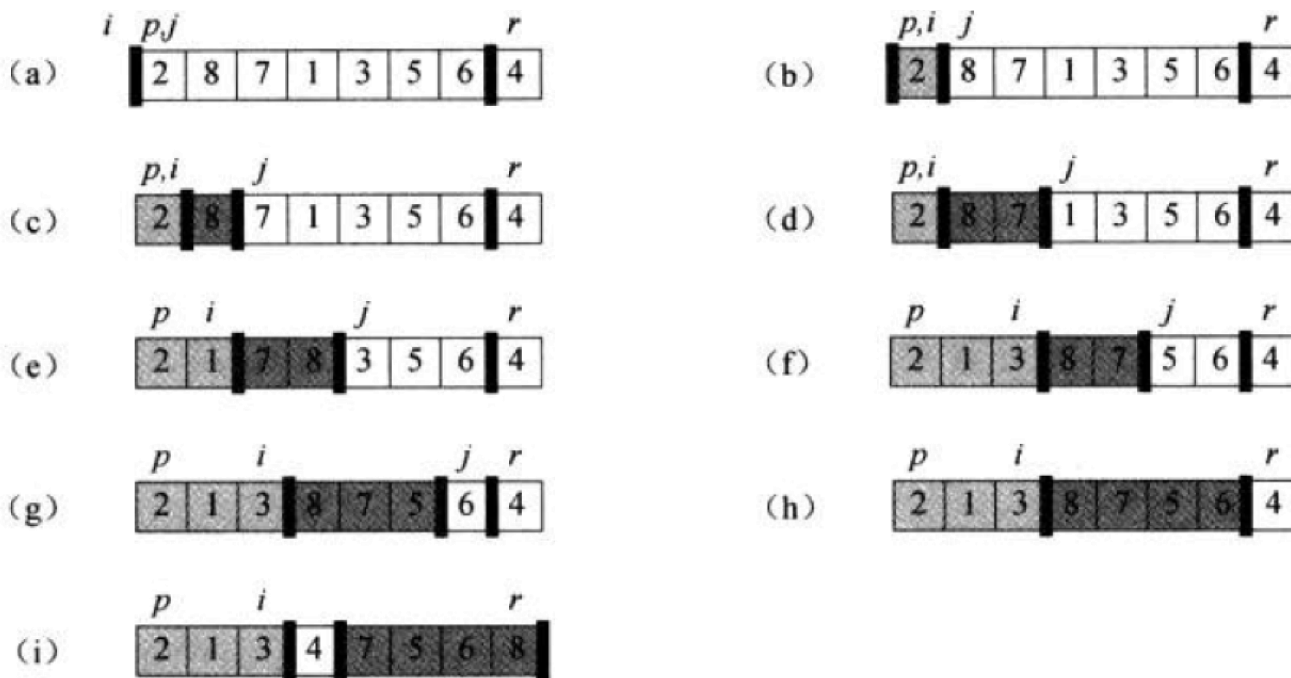


图 7-1 在一个样例数组上的 PARTITION 操作过程。数组项 $A[r]$ 是主元 x 。浅阴影部分的数组元素都在划分的第一部分，其值都不大于 x 。深阴影部分的元素都在划分的第二部分，其值都大于 x 。无阴影的元素则是还未分入这两个部分中的任意一个。最后的白色元素就是主元 x 。(a) 初始的数组和变量设置。数组元素均未被放入前两个部分中的任何一个。(b) 2 与它自身进行交换，并被放入了元素值较小的那个部分。(c)~(d) 8 和 7 被添加到元素值较大的那个部分中。(e) 1 和 8 进行交换，数值较小的部分规模增加。(f) 数值 3 和 7 进行交换，数值较小的部分规模增加。(g)~(h) 5 和 6 被包含进较大部分，循环结束。(i) 在第 7~8 行中，主元被交换，这样主元就位于两个部分之间

分治法求解——快速排序 (续)

■性能分析：划分是否平衡？

➤最坏情况划分（已经有序排列）

$$\begin{aligned}T(n) &= T(n-1) + T(0) + \Theta(n) \\&= T(n-1) + \Theta(n) \\&= \sum_{k=1}^n \Theta(k) = \Theta\left(\sum_{k=1}^n k\right) = \Theta(n^2)\end{aligned}$$

➤最好情况划分（两子问题大小大致相等）

$$T(n) = 2T(n/2) + \Theta(n) = \Theta(n \lg n)$$

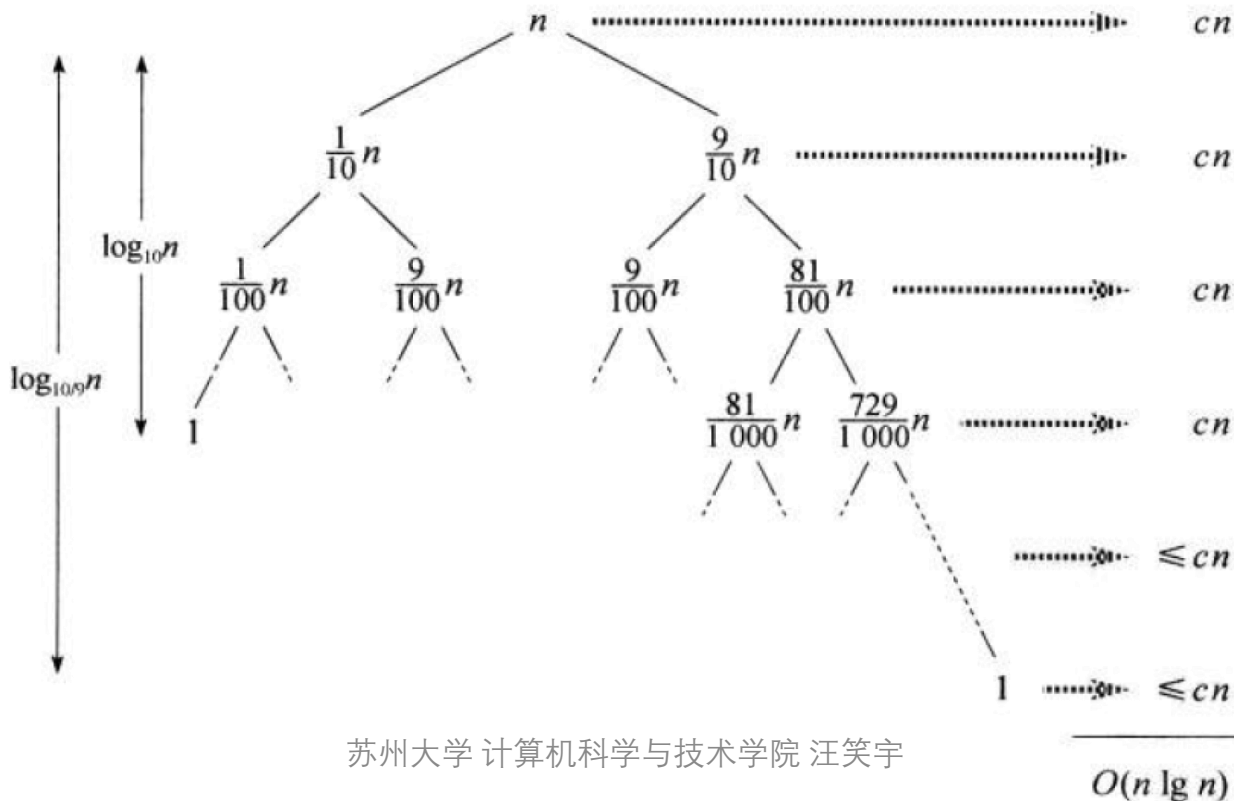
分治法求解——快速排序 (续)

■性能分析：划分是否平衡？

➤平均情况划分（平均情况时间接近于最好情况时间）

- 假设划分总是产生9:1划分

$$T(n) \leq T(9n/10) + T(n/10) + cn$$



分治法求解——快速排序 (续)

■性能分析：划分是否平衡？

➤平均情况划分（平均情况时间接近于最好情况时间）

- 因为任何底大于1的对数与以2为底的对数之间只相差一个常数因子
- 所以任何常数比例的划分，树高仍为 $O(\lg n)$ ，从而快排时间为 $O(n \lg n)$

分治法求解——快速排序 (续)

■随机化版本（教材p100）

- 快速排序的平均性能假定：输入的所有排列是等可能的
- 算法随机化是指：算法行为不仅由输入确定，而且与随机数发生器产生的值有关，强迫输入分布是随机的

```
RANDOMIZED_PARTITION( $A, p, r$ )  
1  $i \leftarrow \text{RANDOM}(p, r)$   
2 exchange  $A[r]$  with  $A[i]$   
3 return PARTITION( $A, p, r$ )
```

- 分析较困难
- 算法非常有效，排序过程中，某次随机选择最坏不会影响总体效果

分治法求解——期望为线性时间的选择算法

■求 n 个元素组成的序列中第 i 小的元素（假设所有元素互异）

```
RANDOMIZED_SELECT( $A, p, r, i$ )
```

```
1 if  $p = r$  return  $A[p]$ 
```

```
2  $q \leftarrow$  RANDOMIZED_PARTITION( $A, p, r$ )
```

```
3  $k \leftarrow q - p + 1$ 
```

```
4 if  $i = k$  return  $A[q]$ 
```

```
5 elseif  $i < k$  return RANDOMIZED_SELECT( $A, p, q-1, i$ )
```

```
6 else return RANDOMIZED_SELECT( $A, q+1, r, i-k$ )
```

```
RANDOMIZED_PARTITION( $A, p, r$ )
```

```
1  $i \leftarrow$  RANDOM( $p, r$ )
```

```
2 exchange  $A[r]$  with  $A[i]$ 
```

```
3 return PARTITION( $A, p, r$ )
```

分治法求解——期望为线性时间的选择算法 (续)

■平均情况： $T(n) = T(n/2) + \Theta(n)$

➤由主方法第三种情况可得： $T(n) = \Theta(n)$

■最坏情况： $T(n) = T(n-1) + \Theta(n)$

➤与快速排序最坏情况相同，为 $\Theta(n^2)$

本章小结

- 递归式（举例：阶乘、斐波那契数列、Ackermann函数、汉诺塔问题、全排列问题）
- 分治法求解：分解、解决、合并
- 求解递归式（计算时间复杂度）：
 - 代入法：猜测解+数学归纳法证明（注意边界条件！）
 - 注意事项：做出好的猜测、细节修正、避免陷阱、变量代换
 - 递归树法
 - 主方法
- 分治法的适用条件：规模缩小到一定程度易解决、最优子结构、子问题解可合并、分解出的子问题相互独立
- 分治法求解应用：二分搜索、最大子数组、矩阵乘法、大整数乘法、快速排序、期望为线性时间的选择算法